K-Means

Paul Hatini

11/2/2020

```
library(mvtnorm)
```

K-Means algorithm

Below I've written a k-means clustering algorithm that aims to partition n two dimensional observations into k clusters in which each observation belongs to the cluster with the nearest centroid. My function takes a two-dimensional dataset, number of centroids (k) and number of desired repetitions (r). Initially, k centroids are chosen randomly from the range of x and y. These centroids are used to train a classifier. The centroids are updated as the means of the observations in each cluster. The process of classification and centroid adjustment is iterated r times.

```
kmeans <- function(k,data,r) {
  centroids <- matrix(nrow = k, ncol = ncol(data))
  for ( i in 1:nrow(centroids)) {
    centroids[i,1] <- sample(data[,1], 1)
    centroids[i,2] <- sample(data[,2], 1)
  }

  error <- matrix(nrow = r, ncol = 1)

  for ( r in 1:r ) {
    # compute distance between n points and k centroids
    distance <- matrix(nrow = nrow(data), ncol = nrow(centroids))
    for ( j in 1:ncol(distance)) {
      for ( i in 1:nrow(distance)) {
        x <- c(data[i,1], data[i,2])
        y <- c(centroids[j,1], centroids[j,2])
        distance[i,j] <- dist(rbind(x,y))
      }
    }

    # update centroids as means of n in k
    nrow(distance[which(distance[,1] > distance[,2]),])
    nrow(distance[which(distance[,1] < distance[,2]),])
    xsum1 <- 0
    xsum2 <- 0
    ysum1 <- 0
    ysum2 <- 0
    for (i in 1:nrow(data)) {
      if (distance[i,1] > distance[i,2]) {
        xsum1 <- xsum1 + data[i,1]
        ysum1 <- ysum1 + data[i,2]
      }
```

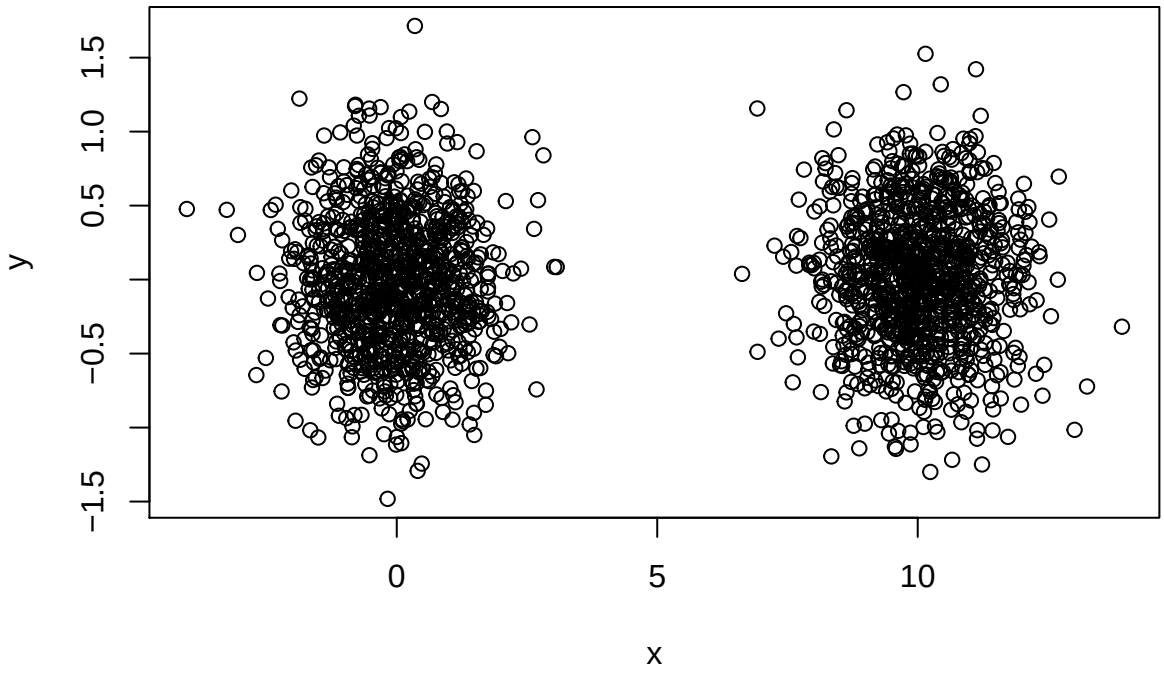
```
      else
        xsum2 <- xsum2 + data[i,1]
      ysum2 <- ysum2 + data[i,2]
    }
    centroids[1,1] <- xsum1/nrow(distance[which(distance[,1] > distance[,2]),])
    centroids[1,2] <- ysum1/nrow(distance[which(distance[,1] > distance[,2]),])
    centroids[2,1] <- xsum2/nrow(distance[which(distance[,1] < distance[,2]),])
    centroids[2,2] <- ysum2/nrow(distance[which(distance[,1] < distance[,2]),])

    error[r,1] <- sum(distance[which(distance[,1] > distance[,2]),2]) + sum(distance[which(distance[,1]

  }
  labels <- matrix(nrow = nrow(data), ncol = 1)
  for ( i in 1:nrow(labels) ) {
    x <- c(data[i,1], data[i,2])
    y1 <- c(centroids[1,1], centroids[1,2])
    y2 <- c(centroids[2,1], centroids[2,2])
    d1 <- dist(rbind(x,y1))
    d2 <- dist(rbind(x,y2))
    if (d1 > d2) {
      labels[i,1] <- "two"
    }
    else
      labels[i,1] <- "one"
  }
  return_list <- list("centroids" = centroids, "labels" = labels, "error" = error)
  return(return_list)
}
```

My function outputs the location of each centroid, the labels of each data point, and the error at each iteration of the algorithm.

Simulated Dataset

```
N <- 1000

# covariance matrix
sigma.mat.ellipse.one <- matrix(c(1,0,0,.2), nrow = 2, ncol = 2)
sigma.mat.ellipse.two <- matrix(c(1,0,0,.2), nrow = 2, ncol = 2)


# mean vectors
mean.vec.ellipse.one <- rep(0,2)
mean.vec.ellipse.two <- c(10,0)


# create matrix of 2 correlated predictors
ellipse.one <- round(rmvnorm(N,mean=mean.vec.ellipse.one, sigma=sigma.mat.ellipse.one),6)
ellipse.two <- round(rmvnorm(N,mean=mean.vec.ellipse.two, sigma=sigma.mat.ellipse.two),6)


x <- c(ellipse.one[,1], ellipse.two[,1])
y <- c(ellipse.one[,2], ellipse.two[,2])
```

```
data <- cbind(x,y)
plot(data)
```



K-Means implementation

```
ellipse.means <- kmeans(2, data, 10)
data <- cbind(data, ellipse.means$labels)
plot(data[,2] ~ data[,1], col=as.factor(data[,3]))
```